VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro snadnou katalogizaci knih
System For a Fast Book Catalogization

2018 **Michal Jež**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Jež**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém pro snadnou katalogizaci knih
System For a Fast Book Catalogization

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit systém, který usnadní katalogizaci domácího knižního fondu. Mezi vybrané požadované schopnosti patří: získávání informací s externích zdrojů jako jsou knihovny a knižní databáze; rychlé snímání klíčových informací s knih pomocí mobilního telefonu a možné automatické zpracování; správa získaných informací a exporty vybraných knih.
Zásady pro vypracování:
1. Zachyťte požadavky na aplikaci – funkční, ale také kvalitativní a omezení (nefunkční).
2. Vytvořte návrh aplikace.
3. Aplikaci naprogramujte
4. Proveďte testování funkčnosti.
5. Postupujte iterativně při vývoji.

Seznam doporučené odborné literatury:

[1] COCKBURN, Alistair, 2000. Writing Effective Use Cases. 1 edition. Boston: Addison-Wesley Professional. ISBN 978-0-201-70225-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kožusznik, Ph.D.**

Datum zadání: 01.09.2017

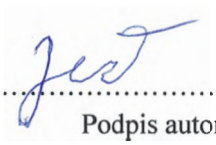
Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě dne 30. dubna 2018



.....

Podpis autora práce

Děkuji panu Ing. Janu Kožusznikovi, Ph. D., za odborné vedení mé bakalářské práce a za cenné rady při jejím zpracování.

Abstrakt

Tato bakalářská práce se zabývá vytvořením systému pro snadnou katalogizaci knih v domácím využití. Jedná se o systém, který uživatelům zefektivní práci při elektronické evidenci svých knih. Práce popisuje, jakým způsobem je katalogizace domácího knižního fondu usnadněna. Cílem je tedy vytvořit systém s několika vlastnostmi, jako jsou snímání klíčových informací pomocí fotoaparátu Android zařízení, automatické dohledání knih na základě těchto informací a následná evidence knih. Mezi další vlastnost patří exportování vybraných knih. Systém by mohl být časem rozšířen na použití ve veřejných knihovnách. Možnosti takového rozšíření jsou uvedeny v závěru celé práce.

Klíčová slova

Android zařízení, aplikace, implementace, ISBN, katalogizace, knihovna, rozšíření systému, systém, technologie, uživatel, XML

Abstract

The Bachelor thesis deals with the creation of a system for easy cataloguing of books in a domestic use. It is a system that allows users to work more efficiently on their electronic books records. This thesis describes the process of simplification the cataloguing of the domestic book collection. Therefore, the goal is to make a system with multiple properties such as capturing the main information using the camera's Android device, automatic book search based on this information, and subsequent book entry. Another feature consists of the export of selected books. The system could be extended to use in public libraries. The possibilities for such an extension are listed at the end of the whole work.

Key Words

Android device, application, implementation, ISBN, catalogization, library, system extension, system, technology, user, XML

Obsah

Seznam použitých symbolů a zkratek 8
Seznam ilustrací 9
ÚVOD 10
1 Již existující systémy 11
1.1 Evidence LSoft 11
1.2 Book Collector 11
1.3 Srovnání systémů 11
2 Použité technologie 12
2.1 Mobile Vision API 12
2.1.1 Face API 12
2.1.2 Barcode scanner API 12
2.1.3 Text Recognition API 12
2.2 JavaServer Faces 13
2.3 REST 13
2.4 Jersey 14
2.5 Jsoup 14
2.6 Apache Tomcat 14
2.7 Hibernate 14
2.8 SQLite 15
2.8.1 Vestavěné zařízení a internet věcí 15
2.8.2 Webové stránky 15
2.8.3 Dočasná paměť 15
2.8.4 Formát přenosu dat 16
2.8.5 Databáze na serverové straně 16
3 Vlastní implementace 17
3.1 Definice pojmů 17
3.1.1 Impresum 17
3.1.2 Tiráž 17
3.1.3 ISBN 17
3.2 Požadavky 18

3.3 Architektura systému ... 19
3.4 Jednotlivé části systému ... 19
3.4.1 Android zařízení ... 19
3.4.2 Server ... 21
3.4.3 Klientská aplikace ... 22
3.5 Použité technologie a stěžejní části systému ... 26
3.5.1 Rozpoznávání ISBN z obrazu ... 26
3.5.2 Vyhledávání informací z externích zdrojů ... 27
3.5.3 Popis třídy ScannedBookBean ... 28
3.5.4 Použitá databáze ... 28
3.6 Vývojové prostředí ... 30
3.6.1 Android Studio ... 30
3.6.2 Eclipse ... 30
4 Rozšíření pro veřejné knihovny ... 32
4.1 Migrace databáze ... 32
4.2 Změny v Android aplikaci ... 32
4.3 Rozšíření klientské aplikace ... 33
5 Ukázky systému ... 34
ZÁVĚR ... 36
LITERATURA ... 37
SEZNAM PŘÍLOH ... 39

Seznam použitých symbolů a zkratek

CGI - Common Gateway Interface

CSV - Comma-separated values

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

HQL - Hibernate Query Language

ISBN - International Standard Book Number

JAX-RS - Java API for RESTful Web Services

JDBC - Java Database Connectivity

JSF - JavaServer Faces

MVC - Model-view-controller

OCR - Optical Character Recognition

PDF - Portable Document Format

QR - Quick Response Code

SOAP - Simple Object Access Protocol

SQL - Structured Query Language

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

XML - Extensible Markup Language

Java EE - Java Platform, Enterprise Edition

Seznam ilustrací

Obr. 1.: Rozpoznávání textu [6] 12
Obr. 2.: Architektura JSF [9] 13
Obr. 3.: Use case diagram 20
Obr. 4.: Stavový diagram 21
Obr. 5.: Use case diagram 23
Obr. 6.: Menu aplikace 23
Obr. 7.: Diagram aktivit 25
Obr. 8.: Titulní stránka aplikace 26
Obr. 9.: Ukázka Android aplikace 34
Obr. 10.: Ukázka serverové části 35
Obr. 11.: Ukázka klientské aplikace 35

ÚVOD

Katalogizace knih pro někoho znamená pár minut, pro jiné je to zdlouhavá práce. Spousta lidí eviduje domácí knihy tím způsobem, že z nich zaznamená potřebné údaje do své databáze jednu po druhé. Tento způsob vyhovuje spíše těm, kteří vlastní menší počet knih. Z daných publikací ale nezjistí všechny informace, například žánr, který se v nich většinou nevyskytuje. V tomto případě musí majitel obsah knihy znát, nebo si jej někde vyhledat. Naopak pro ty, kteří vlastní tisíce publikací a stále jim přibývají nové, je tento způsob velmi zdlouhavý. V takové situaci by jim pomohla efektivnější katalogizace.

V dnešní době existuje několik systémů, které práci při evidenci usnadňují. Například tím, že dokáží samy vyhledat informace na základě parametrů, jako je ISBN. Uživatel tak nemusí zjišťovat údaje tím, že si prolistuje knihu, pouze zadá nezbytné části, které systém sám zpracuje, vyhledá a zařadí do databáze.

Myslíte si, že taková katalogizace už nemůže být jednodušší? V tom se mýlíte, existuje možnost evidence knih bez jejich přenosu k počítači. K tomu nám bude sloužit dnes tolik oblíbené Android zařízení. Právě tímto tématem se budeme v naší práci zabývat.

Bakalářská práce je rozdělena na část teoretickou a praktickou. Teoretická část obsahuje pět kapitol, které se zabývají vysvětlením a popisem systému. Částí praktickou rozumíme samotný systém, který je součástí příloh.

V první kapitole si představíme již existující systémy, které se katalogizacím knih věnují, a navzájem je porovnáme.

V další kapitole si nastíníme nejdůležitější technologie, které při implementaci tohoto systému byly použity.

Třetí kapitola se věnuje samotnému vývoji mého systému. Na úvod definuje základní pojmy, bez kterých se při porozumění práce neobejdeme. Dále popisuje požadavky; tedy co od našeho systému očekáváme, a jeho rozdělení na tři části. Každou z nich si popíšeme. První je Android aplikace, kde se především zaměříme na rozpoznávání textu z obrázků. Další část je serverová, která se nám stará o příjem dat z Android zařízení. Třetí se věnuje klientské aplikaci, u které si detailně představíme vyhledávání informací z externích zdrojů. Na konci této kapitoly si představíme vývojové prostředí, ve kterých byl systém naprogramován.

Následující kapitola se zabývá možným rozšířením systému pro veřejné knihovny.

Poslední kapitola obsahuje snímky pořízené během testování našeho systému.

1 Již existující systémy

1.1 Evidence LSoft

Evidence LSoft [1] je software umožňující katalogizaci filmů a knih. Pro proces katalogizace využívá načítání dat z externích serverů, kterými jsou www.csfd.cz a www.databazeknih.cz. To usnadňuje práci pro uživatele, kteří nemusí veškeré informace psát ručně. Pouze zadají název filmu nebo knihy a software si takové informace sám vyhledá. Mezi důležité vlastnosti programu patří možnost importování informací o knihách a filmech, exportování do formátu PDF a HTML nebo také importování databáze ze souboru CSV.

1.2 Book Collector

Book Collector [2] je aplikace pro přehlednou sbírku klasických, elektronických i audio knih. Program dokáže podle ISBN kódu vyhledat danou knihu a stáhne všechny její informace z internetu. Databázi knih lze navíc snadno sdílet nebo exportovat do formátů XML, HTML nebo CSV. Údaje o knize lze navíc načítat i pomocí čtečky čárových kódů a program spolupracuje i s mobilními zařízeními iPad, iPod, iPhone a mobilním OS Android.

1.3 Srovnání systémů

V této kapitole bylo představeno pár dalších systémů patřících do kategorie elektronických katalogizací knih. Všechny tyto systémy mají společné to, že usnadní práci uživatelům díky automatickému vyhledávání knih. Nyní si je z pohledu právě zmiňovaného usnadnění porovnáme. Zaměříme se na způsob zadávání klíčových informací a následujícímu vyhledávání knih.

Program Evidence LSoft má pro nás obrovskou výhodu v české lokalizaci, to znamená, že pro vyhledávání knih používá český server. Mezi jeho nevýhody patří zejména to, že uživatel nemůže zadat vstupní parametry jinak než ručně. Dále používá pouze jednu databázi knih pro vyhledávání, což znamená, že všechny knihy, které nejsou evidovány na webu www.databazeknih.cz, není schopný vyhledat. Book Collector na rozdíl od softwaru LSoft nabízí možnost skenování knih pomocí čtečky čárových kódů nebo mobilních zařízení. Na druhou stranu nás hodně limituje právě ve vyhledávání knih, kdy vyhledává na zahraničních serverech. To bude pro milovníky českých knih omezující.

2 Použité technologie

2.1 Mobile Vision API

[3] Jedná se o aplikační rámec vyvinutý společností Google. Tento framework poskytuje rozpoznávání objektů z obrázků a videí. Vývojářům umožňuje zajímavé rozhraní v oblasti rozpoznávání obličeje, textů nebo čárových kódu.

2.1.1 Face API

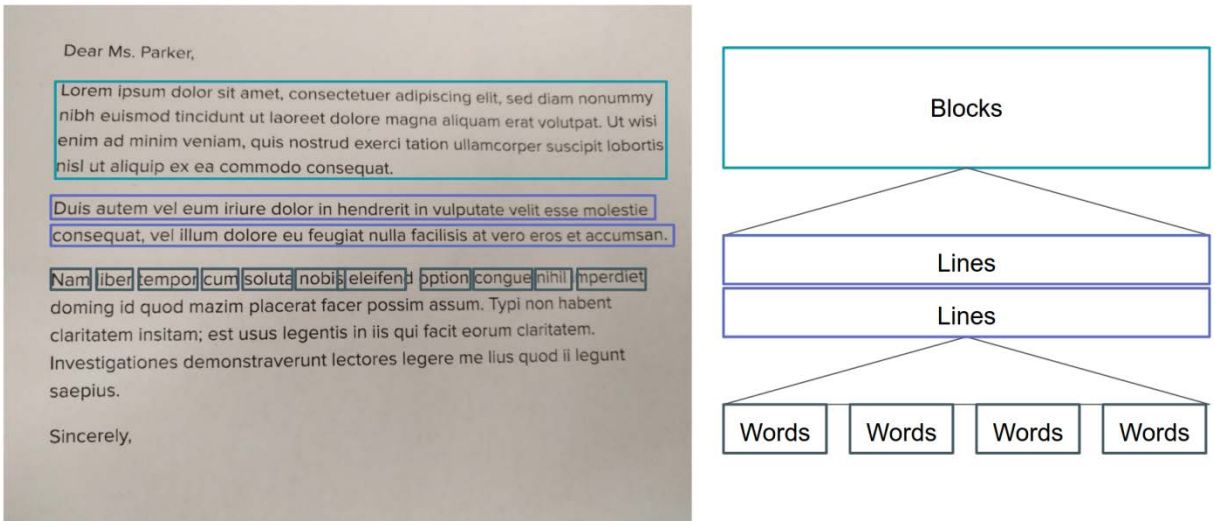
Face API [4] dokáže rozpoznávat obličeje z obrázků, dokonce i z běžících videí. Při detekci se zaměřuje na pozice očí, nosu a úst. Zvlášť zajímavou věcí je, že umí rozpoznat mrknutí a úsměv, díky tomu může být použit například pro ovládání her pomocí mimiky.

2.1.2 Barcode scanner API

Barcode scanner API [5] se používá při klasickém skenování čárových a QR kódů. Oproti běžným skenerům však dokáže skenovat více kódů navzájem, a to i v různých formátech.

2.1.3 Text Recognition API

Text Recognition API [6] slouží k optickému rozpoznávání znaků. Je to proces, při kterém dochází k detekování textu z tištěné podoby do digitalizované. Rozpoznávání od Googlu dokáže zdigitalizovat pouze jazyky, u nichž se používá latinské písmo. Z textu dokáže rozpoznat odstavce, které nazývá bloky, ty dále člení na řádky a slova.

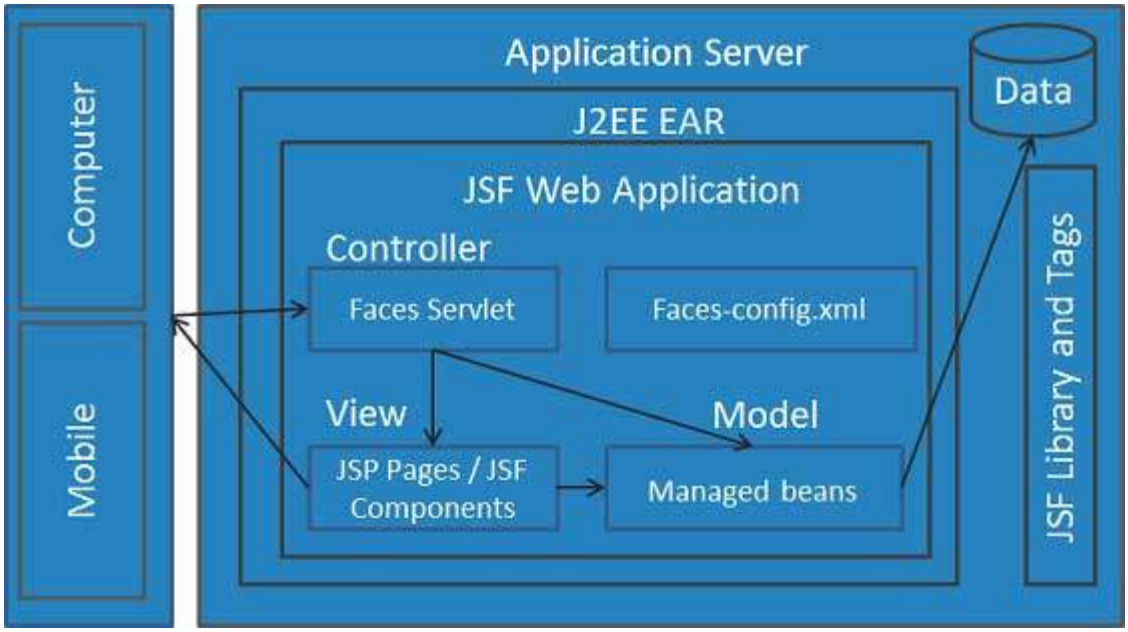


Obr. 1.: Rozpoznávání textu [6]

2.2 JavaServer Faces

JavaServer Faces [7][8][9] je technologie vyvinuta pro vytváření uživatelského rozhraní webové aplikace. Je součástí platformy Java EE, proto můžeme vytvářet webové aplikace bez nutnosti přidávání dalších knihoven. JSF má dvě hlavní funkce. První z nich je generování uživatelského rozhraní HTML jako odpověď na nějaký uživatelský požadavek. Druhou funkcí je reagovat na změny vzniklé uživatelem na stránce a následně generovat jiné uživatelské rozhraní nebo aktualizovat to stávající. Primárním cílem JSF je jednoduché použití. Díky tomu jasně definuje oddělení prezentační vrstvy od aplikační vrstvy, to znamená, že se jedná o technologii založenou na modelu MVC. Tento návrh se hlavně osvědčil při vývoji, kdy každému členovi vývojového týmu dovoluje soustředit se na svůj vlastní proces. Weboví návrháři se soustředí pouze na prezentační vrstvu, zatímco vývojáři se soustředí hlavně na vrstvu aplikační.

Na následujícím obrázku si zjednodušeně popíšeme architekturu JSF. Jedná se o třívrstvou architekturu, která obsahuje Faces Servlet, Managed beans a JSF Pages. Pokud uživatel provede nějakou akci, tak Faces Servlet obdrží informaci o této akci z uživatelského rozhraní. V případě potřeby aktualizace dat přistoupí k Managed beans, který se stará o doménovou logiku a má přístup k datům. Nakonec pošle požadavek na novou nebo upravenou JSF Pages, která zobrazí uživateli výsledek formou HTML stránky.



Obr. 2.: Architektura JSF [9]

2.3 REST

REST [10] je architektonický styl, který nám umožňuje vytvářet, číst, editovat nebo mazat informace ze serveru nad protokolem HTTP. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům. Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). REST je tedy na rozdíl od XML-RPC či SOAP orientován datově, nikoli procedurálně.

2.4 Jersey

Jersey Framework [11][12] je jeden z referenčních implementací JAX-RS. JAX-RS je Java rozhraní, které nám zjednodušuje vývoj RESTful webových služeb. Je to jedno z nejnovějších rozhraní v Javě. Implementace RESTful služby v Javě jsou možné již několik let pomocí servlet. V roce 2008 byla definována nová specifikace JAX-RS, která implementaci RESTful služby zjednodušuje. JAX-RS je framework, který se zaměřuje na použití Java anotací klasických objektů. Nabízí mapování mezi adresou URL a požadavkem HTTP na konkrétní třídu a metodu.

2.5 Jsoup

Java knihovna jsoup [13] nám zjednodušuje parsování stránek HTML. Nabízí nám velmi přívětivé rozhraní pro manipulaci s daty pomocí DOM neboli objektového modelu dokumentu. DOM je API [14][15], které umožňuje obecný standard pro přístup k platným html nebo xml dokumentům. Je založen na struktuře objektů, která se podobá struktuře dokumentů. Umožňuje dívat se na HTML jako na datovou strukturu strom. Jsoup dokáže načíst potřebná data z adresy URL, souboru nebo přímo ze stringu. Jsoup pro vyhledávání v dokumentu podporuje CSS jako syntaxi pro nalezení odpovídajícího prvku.

2.6 Apache Tomcat

Tomcat [16] je open-source Java Servlet kontejner, který je vyvíjen společností Apache Software Foundation. Funguje jako kompletní samostatný webový server. Podporuje statické webové stránky, externí CGI a mnoho dalších doplňků souvisejících s webovými stránkami. Dále umožňuje implementaci Java Servlets a JavaServer Pages (JSP) pro podporu efektivního prostředí Java serveru.

2.7 Hibernate

Hibernate [17][18][19] je řešení objektově-relačního mapování v prostředí Java. Vývoj Hibernate byl zahájen již v roce 2001 Gavinem Kingem. Zpočátku se jednalo o nekomerční open source projekt, který přilákal spoustu vývojářů. Výsledkem bylo používání Hibernate v mnoha tisících aplikacích. Na to musel vývojový tým Hibernate reagovat a zaměstnat mnoho profesionálních vývojářů. Na konci roku 2003 byl software připojen ke společnosti JBoss. Dnes patří mezi jednu z implementací Java Persistence API, díky tomu ho lze použít v jakémkoliv prostředí podporujícím JPA. Stará se o mapování tříd a vlastností jazyka Java do tabulek a sloupců relační databáze. *„Hibernate's design goal is to relieve the developer from 95% of common data persistence-related programming tasks by eliminating the need for manual, hand-crafted data processing using SQL and JDBC."* [1]

Nabízí dva způsoby zápisu metadat pro mapování tříd. Prvním způsobem je zápis do XML souboru, který obsahuje název mapované třídy, seznam jejich atributů a následuje název tabulky a sloupců. Druhým, novějším, způsobem zápisu mapování mezi třídami a tabulkami je přímý zápis do zdrojového kódu, kdy u každé třídy uvedeme pomocí Java anotace název tabulky, a u každého atributu název sloupce. Anotace vznikly za účelem přidání dodatečných informací ve zdrojovém kódu.

[1] KING, Gavin a kol. *Hibernate Getting Started Guide. Preface* [online]. USA: Red Hat, 2012 [cit. 2018-04-24]. Dostupné z: https://docs.jboss.org/hibernate/orm/3.6/quickstart/en-US/html_single/

Během kompilace se pak generují mapovací soubory XML pro Hibernate. Zápis pomocí těchto anotací zpřehledňuje a snižuje celkový počet řádků kódu.

Hibernate má svůj vlastní dotazovací jazyk HQL, který je však téměř stejný jako jazyk SQL. V principu se jedná o rozšířené SQL o objektově-orientované techniky, takže rozumí věcem jako je polymorfismus, dědičnost apod.

2.8 SQLite

SQLite [20] patří mezi relační databázový systém. Jedná se o minimalistický systém relační databáze. SQLite nelze přímo srovnávat s velkými klient/server databázovými systémy jako jsou například MySQL, Oracle, PostgreSQL. Tyto databáze nesou funkci velkých sdílených úložišť. Počítá se u nich s více uživatelskými přístupy, a proto kladou velký důraz na škálovatelnost výkonu. To není cílem naší zmíněné SQLite databáze, ta se snaží poskytnout lokální úložiště dat pro jednotlivé aplikace a zařízení. Zdůrazňuje se u ní jednoduchost, spolehlivost, efektivita a hlavně nezávislost. SQLite tedy nestojí na architektuře klient/server, kde by byla spuštěna jako zvlášť běžící proces, nýbrž je uložena lokálně v podobě souboru s příponou db. Každý takový soubor představuje právě jednu databázi, ve které může být několik tabulek. V následující části práce si popíšeme, ve kterých případech lze vhodně využít tento systém.

2.8.1 Vestavěné zařízení a internet věcí

Jelikož SQLite nevyžaduje složitou správu systému, je vhodné ji použít v zařízeních, která musí pracovat bez odborné podpory. Použití je tedy vhodné v menších zařízeních, která člověk denně používá. Příkladem jsou mobilní telefony, televizory nebo kuchyňské zařízení.

2.8.2 Webové stránky

U webových stránek je velmi důležité, aby si vývojáři dobře promysleli použití SQLite. Ta je totiž vhodná pro použití na webových stránkách s malým nebo středním provozem. Provozem máme samozřejmě na mysli, jak častý je přístup do databáze. SQLite by měla být schopna pokrýt až 100 000 přístupů do databáze za den, což u většiny webových služeb bude dostačující. Hodnota 100 000 přístupů je pouze hrubý odhad, v praxi byl prokázán i desetinásobek.

2.8.3 Dočasná paměť

SQLite lze také použít jako dočasnou paměť pro velké systémy. Představme si systém, kde se data v relačním databázovém systému často nemění. V takovém případě mohou vývojáři použít SQLite jako dočasné úložiště pro data, která se stáhnou z databázového systému a zůstávají v SQLite na klientské straně. To nám umožní snížit zatížení sítě, a v případě výpadku může aplikace na straně klienta pokračovat v provozu.

2.8.4 Formát přenosu dat

Jelikož celá databáze SQLite je uložena v jediném souboru, který je podporován spoustou různých platforem „(...*) it is often used as a container for transfering content from one system to another*,“ [2] může být často použita jako jednotka pro přenos dat z jednoho systému do druhého.

2.8.5 Databáze na serverové straně

SQLite měl úspěch i jako úložiště dat serverových aplikací běžících v datových centrech. Pro přístup k datovému centru je stále zachována architektura klient/server, kdy klienti prostřednictvím sítě posílají požadavky na server, a ten jim vrací potřebné odpovědi. Výhoda tohoto řešení spočívá v použití samostatných databázových souborů SQLite pro různé části databázového systému. Například server může obsahovat pro každého uživatele samostatnou databázi SQLite. To znamená, že na serveru mohou být v jediném okamžiku připojeny tisíce uživatelů, ale každá SQLite databáze je využívána pouze jedním připojením.

[2] *Appropriate Uses For SQLite. Data transfer format* [online]. c2018 [cit. 2018-04-17]. Dostupné z: https://www.sqlite.org/whentouse.html

3 Vlastní implementace

V této kapitole si popíšeme implementaci našeho systému. Na začátek si uvedeme některé důležité pojmy, které jsou podstatné pro pochopení následujícího textu.

3.1 Definice pojmů

3.1.1 Impresum

Impresum [21] je umístěno na začátku knihy přímo za hlavním titulem. Tato část knihy se označuje různými názvy, nejčastěji se jí říká autorská tiráž nebo copyrightová stránka. Dále zde musí být povinně napsáno ISBN a copyrightové záznamy. Ostatní údaje nalezneme v tzv. technické tiráži.

3.1.2 Tiráž

Tzv. technická tiráž [21] se vkládá na poslední stránku knihy, ve výjimečných případech na vnitřní stránku obálky. Obsahuje technické a vydavatelské údaje, které musí být podle zákona v knize uvedeny. Kromě povinných částí se do tiráže zapisují i ty nepovinné. Někdy se v této části knihy uvádí ISBN, které nalezneme i v autorské tiráži.

Mezi povinné údaje tiráže patří:

- Název díla, tedy titul
- Autor a další spolupracovníci, například překladatel
- Nakladatelství a tiskárna
- Rok vydání

V případě, že jde o překlad knihy, tak se navíc uvádí:

- Originální název díla
- Původní nakladatel

Mohou zde být i nepovinné údaje:

- Výtvarník, editor, redaktor, autor doslovu
- Počet stran
- Vydání
- Edice
- Doporučená cena prodejce

3.1.3 ISBN

International standard book number [21], zkráceně ISBN, je celosvětový systém označování knih, který každé knize přiřazuje třináctimístný číselný identifikátor. V každé zemi sídlí Národní agentura ISBN, která se zabývá dodržováním pravidel při přiřazování čísel. Těchto třináct číslic je identických s čárovým kódem, který má každá kniha. Identifikátor se odděluje pomlčkami a řadí se do několika skupin. První skupina se nazývá prefix a obsahuje čísla 978 nebo 979, to označuje, že se jedná o knihu. Poslední skupina je tzv. kontrolní součet, který tvoří jediná číslice. Zbylé skupiny mají

různou délku. U nás a na Slovensku následuje po prefixu, tedy po číslech 978, pomlčka a za ní je identifikátor 80. Celkový součet prefixu a ostatních identifikátorů je vždy dvanáct.

U větších nakladatelství bývají dvouciferné identifikátory, naopak menší nakladatelství mají přiděleny identifikátory šestimístné. V případě, že nakladatel využije všechna přidělená čísla, přiřadí mu Národní agentura ISBN nová. Některé tituly jsou vydávány více nakladatelstvími, v takové situaci musí každý nakladatel přidělit vlastní číselný kód. Tyto identifikátory ISBN jsou v knihách uvedeny v tzv. autorské tiráži, ale často se tisknou i v tiráži technické.

ISBN není oproti jiným státům, jako je třeba Slovensko, povinné. *„Používání ISBN je však důležité pro identifikaci knih a čárové kódy na zadní straně obálky či přebalu jsou zase důležité pro snazší a rychlejší prodej knih.“* [3] Čárový kód může být v knize uveden pouze jeden, na rozdíl od ISBN. Jak už jsme zmínili v předchozí části, v případě vydání více nakladateli je v knize ISBN několik, ale pro čárový kód se využívá pouze jeden identifikátor, na kterém se vydavatelé shodnou. Na knihách je tištěn čárový kód a nad ním se vypisuje dané ISBN, které se odděluje pomlčkami. Čárové kódy se řídí státními normami, které se během let měnily.

Systém ISBN byl zaveden ve Velké Británii v roce 1967, u nás ho nakladatelství užívá až od roku 1989. Původně měl deset znaků a mohl očíslovat jednu miliardu knih. Po několika letech se našly v systému technické chyby a bylo nutné ho upravit. V roce 2007 bylo ISBN přeměněno podle nových státních norem a od 1. 1. téhož roku se píše třináct číslic. Díky této reformě se počet čísel zdvojnásobil.

Ještě před zavedením ISBN byl u nás v 50. letech 20. století vytvořen systém, který jednoznačně označoval české knihy. Identifikátory se skládaly ze tří skupin, které se stejně jako dnes rozdělovaly pomlčkami. Další dvě skupiny vznikly až v pozdějších letech. Dohromady identifikátor obsahoval sedm číslic. Bylo to podobné budoucímu systému ISBN, ale obsahoval několik chyb, které nový systém dokázal odstranit.

3.2 Požadavky

Náš systém vznikl za účelem nabídnout uživatelům snadnou evidenci jejich knih. Pojmem „snadnou“ v názvu informačního systému rozumíme, že uživatel nebude muset informace o každé knize ručně vyhledávat a zapisovat je do systému. Navíc bude mít možnost skenovat klíčové informace pro vyhledávání pomocí mobilního telefonu.

Systém je navržený především pro evidenci domácí knihovny. Důraz při návrhu tohoto systému byl především kladen na jednoduchost a efektivitu při práci. Jelikož systém bude nasazen v domácím prostředí, nepředpokládá se, že by zde docházelo k víceuživatelskému přístupu. Proto pro nás nebyla prioritou škálovatelnost výkonu.

V současné době jistě existuje spoustu takových systému, které usnadňují katalogizaci domácích knihoven. Ve většině případů se však jedná o zahraniční systémy, u kterých mohou nastat problémy s vyhledáváním českých knih, a to z důvodu, že mají implementované vyhledávání pomocí zahraničních databází knih.

[3] PISTORIUS, Vladimír. *Jak se dělá kniha: Příručka pro nakladatele*. 3. vyd. Příbram: Pistorius & Olšanská, 2011. ISBN 978-80-87053-50-8. (str. 53)

3.3 Architektura systému

Celý systém se skládá ze tří částí. První z nich je Android zařízení, které primárně využijeme jako skener při zařazování nových knih. K tomu využijeme fotoaparát, který je dnes součástí každého chytrého telefonu. Android jsme si zvolili, protože se jedná o nejrozšířenější operační systém v přenosných zařízeních. *„Podle posledních průzkumů z tohoto roku obsahuje OS Android více než 80 % mobilních zařízení.“* [4] To znamená, že náš systém bude použitelný pro širokou veřejnost.

Druhou částí systému je server. Jedná se o webový server implementovaný pomocí technologie Apache Tomcat. Server pouze přijímá informace ze zařízení Android a následně je ukládá pro použití v klientské aplikaci.

Poslední část systému tvoří samotná klientská aplikace. Jedná se o webovou aplikaci postavenou taktéž na technologii Apache Tomcat. Může zde vyvstat otázka, k čemu potřebujeme mít klientskou aplikaci implementovanou jako webovou aplikaci. Když se však zamyslíme nad možným rozšířením tohoto systému na knihovny či antikvariáty, máme hned odpověď. V takových situacích je potřeba, aby k systému měli přístup nejenom zaměstnanci, ale i návštěvníci. Pokud by měl být systém někdy v budoucnu rozšířen, nemusí být změněn od základu.

3.4 Jednotlivé části systému

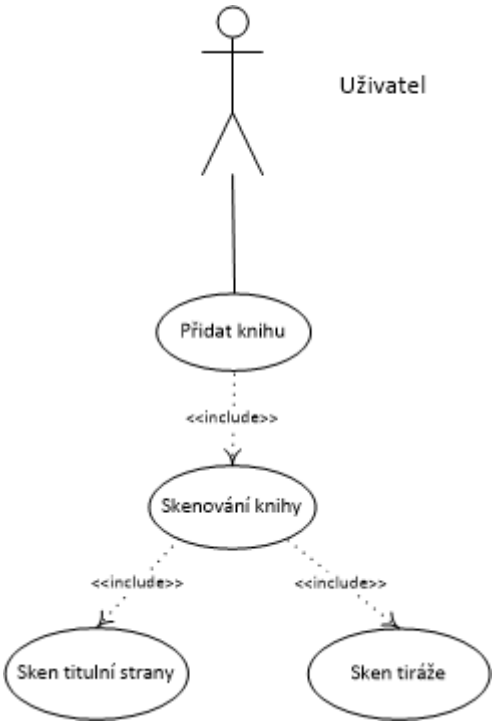
V této kapitole si podrobněji popíšeme jednotlivé části systému podle rozdělení, které bylo uvedeno v předchozí kapitole.

3.4.1 Android zařízení

Jak už je z nadpisu patrné, jedná se o část systému, která běží na platformě Android. [22] Aplikace byla navržena a primárně testována na verzi Android 7.0 (Nougat). Jestliže má být systém využíván do budoucna, musí být pravidelně udržován a aktualizován.

Nyní si vysvětlíme, k čemu nám aplikace pro Android bude sloužit. Chtěli bychom si usnadnit práci spojenou se skenováním nových knih, a právě k tomu tuto aplikaci využijeme. Jelikož potřebujeme, aby katalogizace byla opravdu rychlá a efektivní, je aplikace navržená tak, aby obsahovala pouze nezbytné funkce pro katalogizaci. Tyto funkce si ukážeme na následujícím use case diagramu.

[4] *ANDROID: Vývoj aplikací pro pokročilé i začátečníky* [online]. Praha: GOPAS, 2018 [cit. 2018-04-25]. Dostupné z: https://www.gopas.cz/News/ANDROID--Vyvoj-aplikaci-pro-pokrocile-i-zacatecniky.aspx



Obr. 3.: Use case diagram

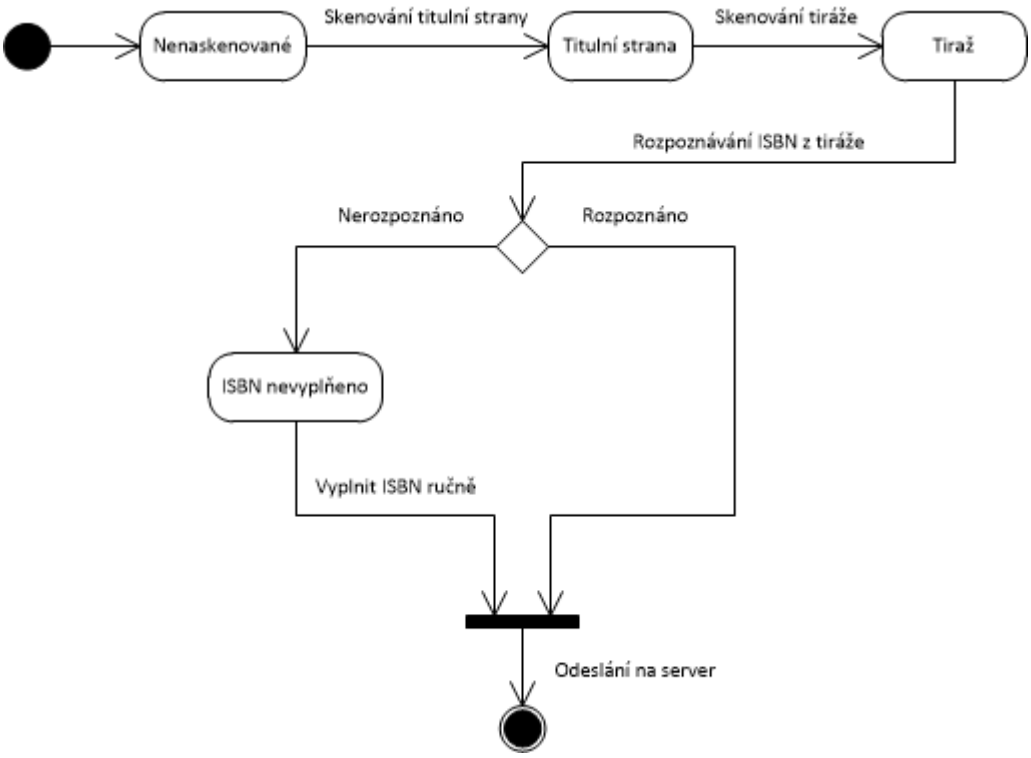
Z předešlého diagramu je patrné, že aplikace pro uživatele nenabízí moc funkcí. Pro náš systém je ale naprosto dostačující. Jednotlivé funkce si společně se stavovým diagramem na obrázku 4 popíšeme.

Po spuštění aplikace může uživatel začít s načítáním nových knih. K tomu mu v aplikaci slouží tlačítko skenovat, po jeho zvolení aplikace spustí fotoaparát na daném zařízení a vyzve uživatele, aby naskenoval titulní stránku. Poté, co uživatel tuto akci provede, aplikace automaticky znovu spustí fotoaparát, aby uživatel mohl naskenovat tiráž. Po naskenování dochází k uložení těchto dvou snímků do paměti.

Přichází fáze zpracování informací z těchto snímků. K tomu nám pomáhá technologie Mobile Vision API, která je popsána ve druhé kapitole. Její detailní implementaci si popíšeme v kapitole 3.5.1. Pomocí Mobile Vision API bude aplikace schopná rozpoznat ISBN z naskenovaných snímků. Po detekování se ISBN v aplikaci objeví v textovém poli.

Spousta knih byla vydána před vznikem dnešního standardu ISBN, kvůli tomu může dojít k situaci, kdy aplikace nebude schopná ISBN poznat. Z tohoto důvodu jsme zachovali možnost, aby uživatel sám mohl ISBN vyplnit. Bohužel najdou se i knihy, které nejenže nemají pevný počet číslic v ISBN, ale dokonce mohou obsahovat i písmena. Tudíž není možné takový vstup od uživatelů nějak ověřit.

Posledním krokem, který musí uživatel udělat, je odeslání nově naskenované knihy na server. Před samotným odesláním je ještě potřeba vyplnit adresu a port serveru. Pokud dojde ke správnému odeslání knihy, server pošle odpověď o správném přijetí a aplikace předá uživateli informaci, že kniha byla úspěšně odeslána.



Obr. 4.: Stavový diagram

Možné rozšíření

Nezbytnou věcí pro funkčnost celé aplikace je přístup k internetu nebo alespoň k síti, na které běží server. V této moderní době se ovšem počítá s tím, že většina uživatelů smartphonů a dalších Android zařízení má přístup k internetu v rámci svých mobilních tarifů nebo alespoň skenuje knihy v prostředí, které je pokryto signálem WiFi. Pro každého však nemusí být přístup k internetu samozřejmostí. Právě kvůli tomu zde tento nedostatek uvádíme. Aplikace je navržená tak, že snímky zachycené fotoaparátem ukládá do externí paměti telefonu. Možným rozšířením by mohlo být neukládat si pouze snímky poslední naskenované knihy, ale nechat je ukládat do nějaké databáze spolu s rozpoznaným ISBN. To by však znamenalo mít dostatečně velkou kapacitu úložiště. V okamžiku naskenování všech knih nebo přístupu k internetu by pak mohly být poslány na server.

3.4.2 Server

V této kapitole se detailněji podíváme na druhou část našeho systému, kterou je server. Server je jediná část našeho systému, se kterou uživatel moc do styku nepřichází. To však neznamená, že by tato část byla nepodstatná, ba naopak. Jeho hlavním úkolem je shromažďovat informace přijaté ze zařízení Android.

Server uživatelům z hlediska uživatelského rozhraní poskytuje pouze jedinou stránku. Na této stránce je uvedena informace o adrese a portu, na kterých server běží. Tyto údaje jsou jediné, které uživatel potřebuje znát, protože je musí zadat do Android zařízení pro správné odeslání svých knih.

Nyní se ponoříme do implementace našeho serveru. Ten pro náš systém musí splňovat následující požadavky. Musí poskytnout službu našemu Android zařízení, aby měl kam poslat své údaje o knize, a následně tyto údaje zaznamenat pro pozdější práci v klientské aplikaci.

K příjmu informací z Android zařízení jsme využili služby REST, která nám umožňuje komunikaci mezi dvěma zařízeními pomocí protokolu HTTP. Pro náš systém jsme k implementaci služby REST použili již zmíněnou technologii v druhé kapitole, a tou je Jersey. Server na své straně očekává vstup od klienta v podobě ISBN obsaženého v textovém řetězci a dvou datových proudů, v nichž jsou uložené snímky pořízené v Android zařízení. Samozřejmě server také vrací relevantní odpověď. Jak jsme si již dříve uvedli, mohou nastat situace, kdy kniha neobsahuje ISBN, proto server ověřuje pouze to, zda klient správně odeslal naskenované snímky, a pokud tomu tak je, posílá odpověď o správném přijetí.

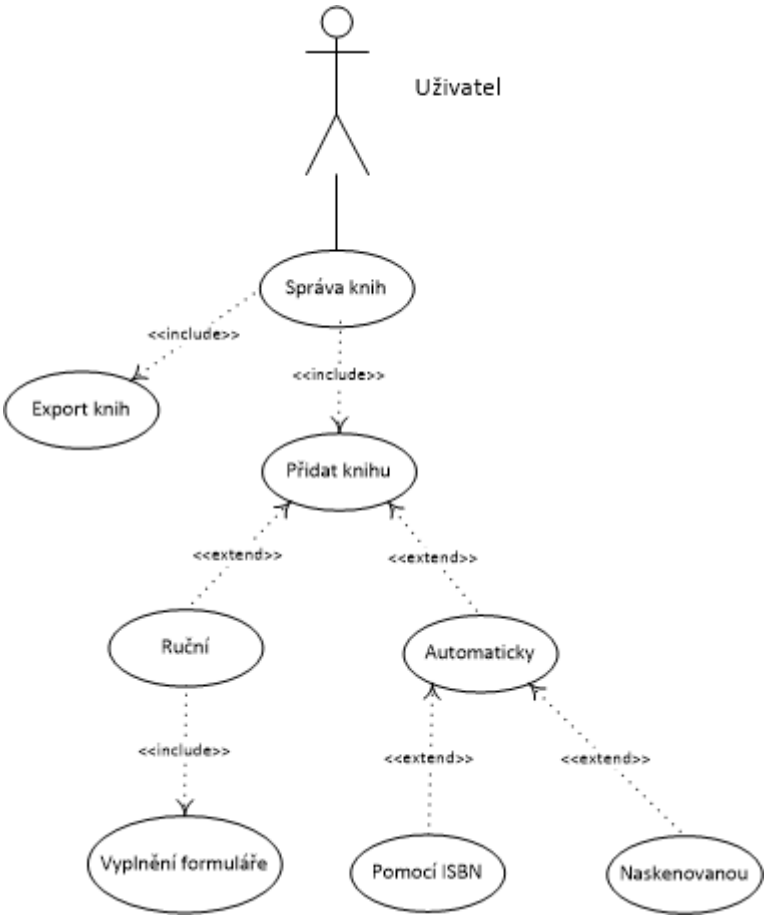
3.4.3 Klientská aplikace

Klientská aplikace je třetí a zároveň poslední část našeho vyvíjeného systému. Jedná se o webovou aplikaci postavenou na technologii JavaServer Faces. Důvodem použití této technologie pro nás bylo, že dokáže oddělit uživatelské rozhraní od logiky aplikace. Pro přístup k aplikační logice se využívají JavaBeans. To jsou Java třídy, které přes své instance předávají obsah, který se má zobrazit v uživatelském rozhraní.

Konfigurace těchto tříd probíhá v souboru faces-config.xml. V tomto souboru je nakonfigurováno mapování všech JavaBeans na jména, která se následně používají přímo jako odkazy na data v HTML kódu. Dále zde můžeme nastavit, jak často se má instance daného JavaBeans vytvářet.

Výhodou oddělení uživatelského rozhraní a logiky aplikace je, že klient nemusí mít nainstalované žádné speciální prostředí. K zobrazení obsahu mu stačí pouze prohlížeč. Pro tvorbu webových stránek, tedy jejich textové části, byl použit značkovací jazyk XHTML doplněný o kaskádové styly, které nám umožňují měnit vzhled prvků napsaných v HTML.

Z pohledu uživatele se určitě jedná o část systému, která mu nabídne nejvíce funkcí. Ty nejdůležitější si ukážeme na následujícím use case diagramu.



Obr. 5.: Use case diagram

Na první pohled si můžeme všimnout, že se nejedná o nikterak složitý systém. Ten má spíše evidenční povahu. V následujících řádcích si postupně rozebereme implementaci naší aplikace.

Obr. 6.: Menu aplikace

Po zadání adresy webové aplikace nás webový prohlížeč přesměruje na titulní stránku. Na této stránce nalezneme knihy, které již máme zařazené ve své evidenci. Tyto knihy můžeme jednoduše editovat. Dále máme možnost je exportovat do souboru CSV. Pokud se podíváme do menu naší aplikace, zjistíme, že obsahuje položky, jako jsou výpis a naskenované knihy. Poslední položkou je přidat knihu. Tlačítkem výpis nás systém pouze přesměruje na titulní stranu. Při přesměrování na stránku s naskenovanými knihami se posílá požadavek na obsah do třídy ScannedBookBean, její implementace je popsána v kapitole 3.5.3. V této třídě jsou nejprve záznamy načteny z databáze,

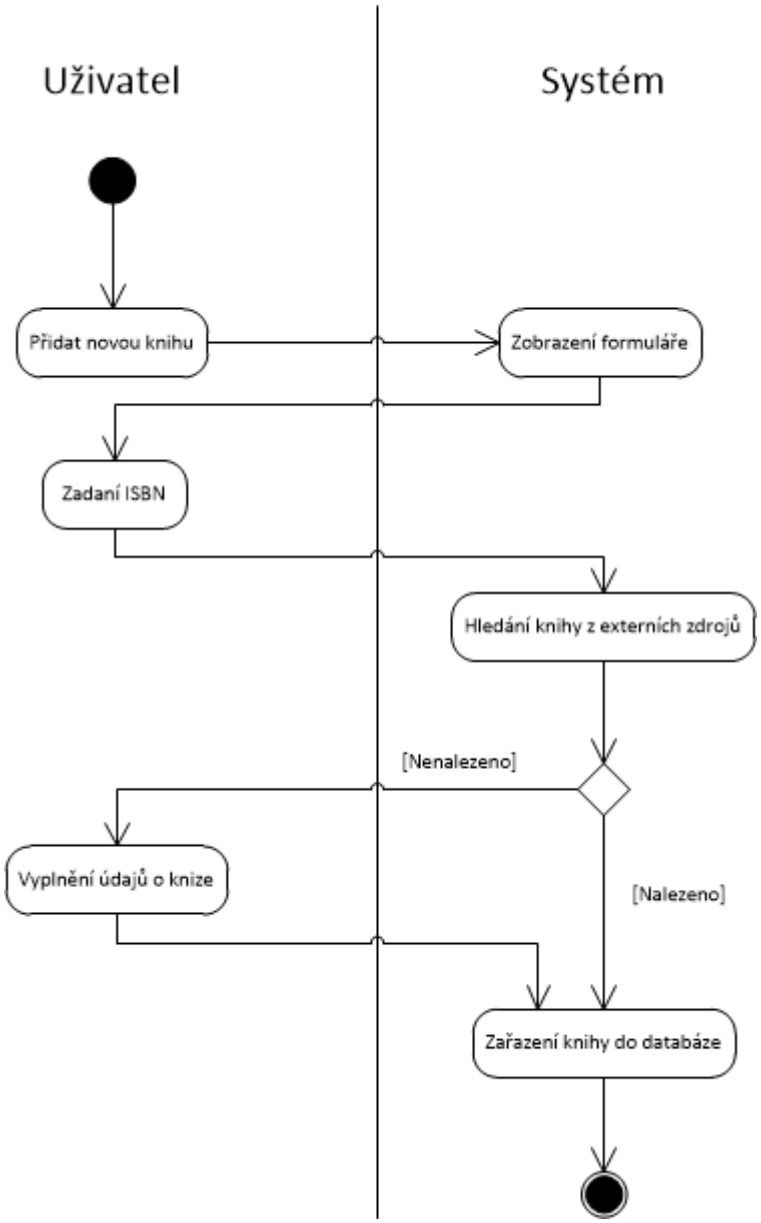
a následně dochází k automatickému vyhledávání informací o knihách z externích zdrojů, které je detailně rozebráno v kapitole 3.5.2.

Po úspěšném načtení a vyhledávání knih se výsledek uživateli objeví v jeho prohlížeči. Zobrazí se mu stránka, na které nalezne tabulku, jež bude obsahovat tyto sloupce: ISBN, název, autor, žánr, vydavatel, počet stran a své dva snímky pořízené při skenování. Snímky jsou samozřejmě zmenšené do podoby náhledu. Poté, co klikneme na snímky, dochází k jejich zvětšení. K implementaci této vlastnosti byl použit JavaScript. Jedná se o objektově orientovaný skriptovací jazyk. Jeho nejčastějším použitím jsou právě webové stránky. Na rozdíl od používání našeho JSF je JavaScript napsán přímo v HTML kódu, což znamená, že se nespouští na straně serveru, ale na straně klienta.

Dále má uživatel možnost si zobrazit pouze knihy, které byly či nebyly vyhledané. U každé knihy je k dispozici odkaz, kterým se uživatel dostane přímo na editaci jednotlivé knihy. Když uživatel zvolí tento odkaz, je přesměrován na stránku, na níž se zobrazí formulář pro vyplnění potřebných údajů. Jelikož se může stát, že ISBN bude posláno z telefonu špatně, může ho zde uživatel opravit a zkusit knihu znovu vyhledat. V případě, že by ani oprava identifikačního čísla knihy nepomohla, nezbývá mu nic jiného, než se pokusit danou knihu sám vyhledat a potřebné údaje doplnit.

Nyní se podíváme na poslední fázi zpracování naskenovaných knih, kterou je zařazení do konečného katalogu. Knihy můžeme zařadit do katalogu po jedné, k tomu nám slouží tlačítko uložit vedle každé knihy. Existuje však i rychlejší možnost, kterou je tlačítko uložit v menu. To nám zařadí všechny označené knihy.

Poslední položkou v menu naší aplikace je přidat knihu. Funkci této stránky si ukážeme na následujícím diagramu a v krátkosti popíšeme.

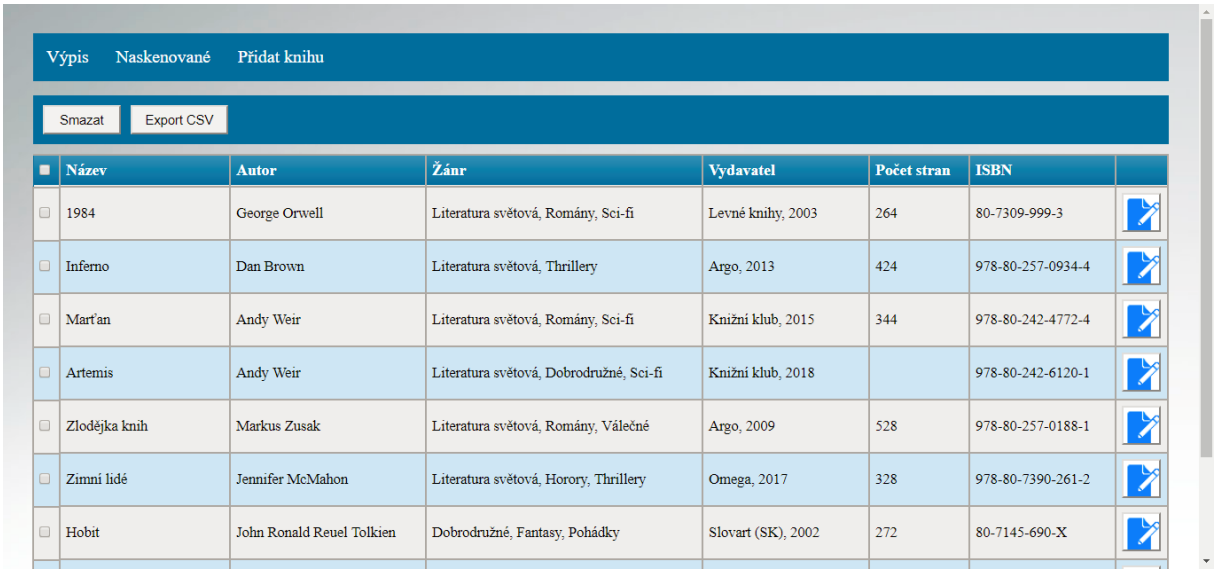


Obr. 7.: Diagram aktivit

Jsme si zcela vědomi, že mohou nastat situace, kdy způsob naší snadné katalogizace bude kontraproduktivní. Předpokládáme, že v situaci, kdy uživatel začne používat náš systém, bude mít doma poměrně velkou knihovnu, a tudíž využije všech možností, které mu náš software nabízí. Představme si naopak situaci, že uživatel už má zkatalogizované všechny knihy a přikoupí si další. V tento okamžik může být proces s mobilním zařízením naopak pomalejší než při katalogizaci většího počtu knih. To je důvod, proč jsme do webové aplikace vložili možnost přidání knihy pomocí klávesnice. Samozřejmě zde zůstává možnost, že uživatel napíše ISBN a systém se pokusí sám knihu vyhledat tak, jak je zobrazeno na obrázku Diagram aktivit. Po vyplnění potřebných údajů je kniha zařazena do katalogu knih.

Možné rozšíření

Na předchozích stranách jsme popsali všechny funkce, které nám aplikace nabízí. Nyní je čas podívat se také na to, co by náš systém mohl dále obsahovat. Naším cílem bylo, co možná nejvíce urychlit práci při katalogizaci knih. Pokud se však podíváme na uživatelské rozhraní, jistě by nás napadla spousta věcí, které by zde mohly být doimplementované. Některé z nich si zde popíšeme.



Obr. 8.: Titulní stránka aplikace

Při pohledu na titulní stranu, jak ji vidíme na obrázku 8, bychom určitě ocenili, kdybychom mohli své knihy nějakým způsobem filtrovat. Knihy se nám do tabulky zobrazují v pořadí, v jakém jsou uloženy v databázi. Určitě by stálo za to doplnit alespoň možnost seřazení podle různých atributů, například názvu knihy nebo jména autora. Dále bychom zde mohli mít textové pole, do kterého bychom zadali klíčové slovo a systém by se podle něj pokusil najít co možná nejvíce relevantní záznam. Stránka obsahuje možnost exportování všech knih do souboru CSV, takže by zde mohla být možnost exportu i do jiných formátů jako jsou například XML, PDF. Z důvodu přehlednosti a zrychlení načítání stránky bylo implementováno stránkování tabulky. Počet knih na stránce je pevně nastaven na maximální hodnotu 30. Některým uživatelům to může připadat moc, jiným zase málo. Proto další věcí, kterou bychom mohli stránku rozšířit, je přidat možnost, aby si každý uživatel sám mohl nastavit počet knih na jedné straně.

3.5 Použité technologie a stěžejní části systému

3.5.1 Rozpoznávání ISBN z obrazu

Pro proces rozpoznávání ISBN jsme využili technologii Mobile Vision API. Z této knihovny budeme využívat pouze Text Recognition API, který poskytuje naší aplikaci takzvané OCR neboli Optical character recognition. Je to způsob rozpoznávání a digitalizace textu z naskenovaného obrázku. To se zdá být jako perfektní řešení pro náš systém. Existuje však i negativní vlastnost této technologie. Jelikož u rozpoznávání textu hodně závisí na kvalitě obrázku, může se stát, že Text Recognition

nebude schopný rozpoznat daný text, nebo zamění některé znaky za jiné. Z tohoto důvodu by bylo u novějších knih lepším řešením používat spíše Barcode scanner API, který by skenoval ISBN pomocí čárových kódů.

Důvodem, proč jsme si vybrali právě Text Recognition API, je, že každý z nás má jistě doma starší knihy, které byly vydány dříve, než vznikl standard pro ISBN s čárovými kódy. Konverze textu z obrázku je proces, který zabere nějaký čas, zvlášť u starších zařízení. Kvůli tomu je rozpoznávání textu implementováno pouze na druhý snímek. V takovém případě je pro uživatele důležité, aby skenovali titulní stranu a tiráž ve správném pořadí tak, jak je k tomu vyzve i samotná aplikace.

Po použití Text Recognition API dostaneme textový řetězec všeho, co bylo možné na snímku rozpoznat. Poslední částí při zpracování textu je vyhledávání ISBN. K tomu využijeme API v balíčku regex pro práci s textem skrze regulární výrazy. V aplikaci je použitý následující regulární výraz "[0-9]{1,3}+[-]{1}+[[0-9]*[-]]*". Cílem tohoto výrazu je najít textový řetězec, který začíná jednou až třemi číslicemi, po nichž následuje spojovník a opět číslice. To vše se může několikrát opakovat. Důvodem, proč není regulární výraz navržen podle standardu ISBN, jsou právě již zmíněné starší knihy, u kterých může být větší nebo menší počet číslic.

3.5.2 Vyhledávání informací z externích zdrojů

Nyní se podíváme, jak bylo řešeno automatické vyhledávání knih. V této práci byl již několikrát zmíněn pojem ISBN. Teď se tedy konečně dostaneme k tomu, proč jsme kladli takový důraz na tento termín. V dnešní době existuje nemálo webových stránek, jejichž obsahem je určitá databáze knih. Může se jednat o stránky různých prodejců knih, knihoven a antikvariátů. Většina těchto webových stránek obsahuje vyhledávací pole, aby usnadnily návštěvníkům vyhledávání pro ně přínosného obsahu. Tak jako při vyhledávání čehokoliv i u knih hledáme podle klíčových slov. Těmi jsou například název knihy, autor nebo právě zmiňované ISBN. Každá kniha má své vlastní jedinečné ISBN, a to je velká výhoda oproti ostatním údajům. To znamená, že vyhledávání podle něj nám buď přinese výsledek v podobě jediné knihy, nebo knihu nemají evidovanou a tím pádem bude výsledek nulový. Když tedy jako návštěvníci takových stránek můžeme své knihy takto jednoduše vyhledat, proč to nepoužít v naší aplikaci a usnadnit tak práci našim uživatelům. A právě to jsme udělali. Za účelem vyhledávání knih pro náš systém jsme se rozhodli použít následující tři české webové servery, které se zabývají tematikou knih.

Prvním z nich je webová stránka www.databazeknih.cz. [23] Co nám tato stránka nabízí? *„Kromě rozsáhlé knižní databáze zde najdete hodnocení knih, názory uživatelů, informace o autorech, povídkách, aktuality z literárního světa a mnoho dalšího.“* [5] Druhým takovým serverem je (nám v regionu asi známá) knihovna města Ostravy s odkazem https://tritius.kmo.cz/Katalog/. Jako poslední možnost pro naše vyhledávání jsme použili databázi Národní knihovny České republiky, ke které jsme využili tento odkaz https://aleph.nkp.cz/F/?func=scan&scan_code=SBN.

Teď si ukážeme, jak jsme v naší aplikaci implementovali vyhledávání knih. V adresáři webové aplikace nalezneme balíček s názvem com.search.service. V tomto balíčku jsou obsaženy pouze

[5] FIALA, Daniel. *Databáze knih: Váš knižní svět* [online]. Polná: Databazeknih.cz, 2018 [cit. 2018-04-26]. Dostupné z: https://www.databazeknih.cz/

rozhraní a třídy, které jsou pro vyhledávání nezbytné. Tak jako máme tři webové stránky, ze kterých čerpáme informace o knihách, tak jsme si vytvořili tři třídy pro každou stránku zvlášť. Dále zde nalezneme jedno rozhraní SearchService, které obsahuje pouze jednu metodu, a tou je SearchBook. Všechny tři třídy toto rozhraní implementují. Již zmíněné stránky používají k vyhledávání dotazovací HTTP metodu GET. To nám ulehčilo práci, neboť GET využívá takzvaných URI dokumentů, takže nám stačí pouze měnit parametry v adrese URL.

K nalezení potřebných informací jsme využili knihovnu jsoup. Ta nám dokáže ze struktury HTML kódu vytvářet strom objektů, a tím nám umožnuje intuitivně parsovat zdrojový kód stránky. My jsme pro naši aplikaci využili pouze dvě třídy, které jsoup knihovna obsahuje. První z nich je třída Document, která nám reprezentuje již zmíněný strom ze staženého zdrojového kódu webové stránky. Další třídou je Element, již instance jsou jednotlivé uzly v tomto stromu. Pak už stačí pouze projít celý strom a pomocí metod, které má Element implementované, najít potřebné atributy a ty si uchovat. To vše je obsaženo v metodách SearchBook u všech tříd v balíčku com.search.service.

3.5.3 Popis třídy ScannedBookBean

Tato třída obsahuje jednu statickou proměnnou typu List, která se inicializuje při prvním vytváření instance této třídy. Je to seznam objektů třídy ScannedBook. Instance této třídy obsahuje atributy, jako jsou název knihy, autor, žánr, vydavatel a počet stran, a k tomu atributy, které jsou mapovány z databáze skenovaných knih.

Nyní se podíváme, co dalšího se děje v konstruktoru třídy ScannedBookBean. Nejdříve si z databáze vybereme všechny naskenované knihy a načteme je do našeho seznamu. Následuje část vyhledávání knih z externích zdrojů. K tomu využijeme balíček com.search.service, který jsme si již popsali. Právě jsme se dostali k důvodu, proč jsme si pro všechny třídy, které se nám starají o vyhledávání, vytvářeli rozhraní SearchBook. Vytvořili jsme si totiž další seznam, který nese objekty typu SearchBook, a do něj jsme přidali všechny třídy, které nám zajišťují vyhledávání. V tento okamžik už můžeme nechat všechny naskenované knihy projít procesem vyhledávání. Ten je nastavený tak, že pokud bude kniha nalezena na kterémkoliv serveru, nebude dál hledána na ostatních.

Teď si však musíme uvědomit, že pokud máme stovky naskenovaných knih, nebude vyhledávání otázkou vteřiny. Nevýhodou je, že se instance třídy ScannedBookBean vytváří po každé aktualizaci stránky. To znamená, že proces vyhledávání knih by byl prováděn neustále dokola, což není úplně vhodné s ohledem na čas strávený vyhledáváním. To je hlavním důvodem, proč máme seznam naskenovaných knih jako statickou proměnnou. Díky tomu se vyhledávání každé knihy uskuteční pouze jednou.

3.5.4 Použitá databáze

Serverová část

Uchování přijatých dat jsme se rozhodli řešit pomocí relační databáze, a to konkrétně SQLite. Díky tomu, že potřebou serveru je pouze vkládat do databáze jeden textový řetězec, který reprezentuje ISBN a dva naskenované snímky, není nutné pro takový server vytvářet složitou strukturu databáze. Z tohoto důvodu jsme vytvořili pouze jedinou relaci se třemi atributy. K tomu, abychom mohli snímky z fotoaparátu uložit do SQLite, je potřeba konverze obrázku do datového typu BLOB. Problém tohoto

datového typu je, že databáze neví, jak jeho obsah interpretovat. Ve výsledku to znamená, že obrázek sice do databáze vložíme, ale pro správné získání zpět budeme muset znát jeho původní formát. To je důvod, proč jsme do relace museli přidat další dva atributy, které nesou informaci o tom, jaký typ souboru byl do formátu BLOB konvertován. Dohromady relace obsahuje pět atributů.

Pro objektově-relační mapování jsme využili Java frameworku Hibernate. V adresáři projektu jsou obsaženy soubory XML, pomocí kterých je Hibernate nakonfigurován. Jedním ze souborů je hibernate.cfg.xml, který je důležitý pro správný běh frameworku. V tomto souboru je popsáno s jakou databází má Hibernate spolupracovat. Existuje mnoho vlastností, které si zde můžeme nastavit. My si však uvedeme pouze ty, které jsou podstatné pro správnou funkčnost Hibernate.

Jednou z vlastností je dialect, tato vlastnost nám umožňuje generování příslušných SQL dotazů pro vybranou databázi. Další vlastností je connection.driver_class, ve kterém je uvedené rozhraní pro přístup k relační databázi. Poslední důležitou vlastností je connection.url. Jak již dříve bylo popsáno, SQLite je databáze, která je uložena v jediném souboru. Proto je jako connection.url zadána cesta k databázovému souboru, který je lokálně uložen na stroji, na němž server běží. Pokud bychom používali některou z databází stojících na principu klient/server, pak bychom místo cesty k souboru udávali adresu databázového serveru. Mezi další vlastnosti, které mohou být v konfiguračním souboru definované, jsou například uživatelské jméno a heslo pro přístup do databáze nebo takzvaný pool size, který omezuje počet klientů čekajících ve frontě na připojení k databázi.

Náš systém obsahuje ještě další XML soubor, ve kterém je obsaženo objektově/relační mapování třídy nesoucí informace o přijaté knize. Je zde definované jednoduché mapování, kde prvním atributem je primární klíč definovaný jako číslo typu integer s automatickou inkrementací. Následuje ISBN, které je definováno jako text o maximální délce nastavené na 100 znaků. Další dva atributy jsou typu BLOB pro obrázky. Ty jsou následovány posledními dvěma atributy nesoucí informaci o typu souboru.

Webová aplikace

Tak jako u serverové části i zde je hlavním představitelem databázového úložiště SQLite. Přístup a mapování k ní je opět zajištěno pomocí frameworku Hibernate. Jedinou změnou oproti serveru je kromě mapování třídy naskenovaných knih také mapování další třídy, kterou je Book. Instance této třídy již budou sloužit jako katalog naší knihovny. Book má stejné atributy jako třída ScannedBook s výjimkou snímků titulní strany a tiráže, protože není žádný důvod k tomu, abychom je ve výsledném katalogu evidovali. Metody, které využíváme pro přístup do databáze, jsou obsaženy ve třídě Database, která je součástí našeho balíčku com.database.

Teď se podíváme, jakým způsobem bude výsledný katalog uložen v naší databázi. Nejdříve si připomeňme všechny atributy, které chceme u každé knihy v katalogu evidovat. Těmi jsou název knihy, jméno autora, žánr, vydavatel, počet stran a ISBN. Pro uchování těchto informací jsme si v databázi vytvořili relaci Book. Pokud se podíváme na jednotlivé sloupce tabulky Book, zjistíme, že atribut autor není atomický. Atomický atribut znamená, že je dále nedělitelný. Kdybychom chtěli, aby naše relace splňovala první normální formu, museli bychom jméno autora rozdělit na dva atributy, a to jméno a příjmení. Zde se dostáváme k problému vyhledávání informací z externích zdrojů.

Naším cílem je poskytnout uživatelům co možná největší spolehlivost při vyhledávání informací knih, proto jsme získávání těchto informací implementovali pomocí více webů s knižní

tematikou. Problém nastává, že každá knihovna, popřípadě databáze knih si svůj katalog vede odlišně. Příkladem je právě zmiňovaný autor. Některé knihovny definují autora ve formě jméno a příjmení, jiné začínají příjmením a další jména oddělují čárkou. Na tomto příkladu jasně vidíme, že pokud bychom chtěli mít relaci alespoň v první normální formě, museli bychom buď nechat na uživateli, aby si sám autora zapsal, nebo používat pouze jeden web pro vyhledávání, čímž bychom se ochudili o spolehlivost při vyhledávání.

Z předchozích řádků je nám asi jasné, že systém z pohledu databáze nese velké nedostatky především v normálních formách. Díky atributu, ve kterém je uveden autor, nám databáze nesplňuje ani první normální formu. To nám samozřejmě přináší problémy v podobě nadbytečných dat neboli redundance.

3.6 Vývojové prostředí

Uvedli jsme si několik možností, jak by mohl být náš systém v budoucnu rozšířen. Z tohoto důvodu by bylo vhodné také uvést, v jakých vývojových prostředích jsme systém programovali. Pro práci jsme využili dvě prostředí, která si v následující části textu zmíníme.

3.6.1 Android Studio

Jedná se o oficiální integrované vývojové prostředí pro vývoj aplikací operačního systému Android založené na technologii IntelliJ IDEA. [24][25] Vývojové prostředí je možné stáhnout pro operační systém Windows, macOS a Linux. Android studio bylo oficiálně představeno společností Google v roce 2013. Do této doby probíhal vývoj aplikací pro Android ve vývojovém prostředí Eclipse.

Android studio nám nabízí jednotné prostředí pro vývoj aplikací pro všechna zařízení běžících na operačním systému Android. Dále nám poskytuje několik funkcí, jakými jsou například možnost rozsáhlého testování vyvíjené aplikace, spravování verzí při vývoji pomocí GitHUB nebo podpora Google Cloud. Důležitou vlastností je, že obsahuje emulátor, který nám umožňuje běh programu pod jinou platformou, než byl původně napsán. To znamená, že vývojářům urychlí práci tím, že pro každé ověřování funkčnosti vyvíjené aplikace ji nemusí instalovat do zařízení se systémem Android.

Když se podíváme na strukturu vytvářeného projektu, zjistíme, že se skládá ze tří složek. První z nich je manifest, ve kterém je obsažen soubor manifest.xml, kde jsou popsány základní informace, jako jsou balíčky, služby, activities. Další složkou je Java, ta je naplněna všemi soubory, které obsahují Java kód. Poslední z nich je složka res. Na rozdíl od složky Java obsahuje nejčastěji konfigurační soubory XML.

3.6.2 Eclipse

Jedním z nejpoužívanějších vývojových prostředí pro programování v jazyce Java je Eclipse [26][27], které začalo jako technologie vedená společností IBM. Společnost tím chtěla snížit velké množství nekompatibilních vývojových prostředí, které v tu dobu nabízela. Právě vývojové prostředí, jako byly IBM VisualAge for Smalltalk a IBM VisualAge for Java, poskytli základ k tvorbě Eclipse. Toto prostředí je z velké části napsané v Javě, a proto jeho primárním použitím je právě vývoj aplikací pro tento jazyk. To však neznamená, že je určen pouze pro něj.

Eclipse nabízí spoustu možných rozšíření pomocí pluginů. Mezi ně patří mimo jiné podpora dalších jazyků, kterými jsou například C, C++, PHP, Javascript. Pro náš systém bylo důležité, že podporuje vývoj pro Tomcat, GlassFish a mnoho dalších serverů. Často je schopný požadovaný server sám naistalovat. Nabízí taktéž možnost za běhu ladit aplikace spuštěné na serveru. Za dobu, co Eclipse existuje, vznikla celá řada jeho verzí. My jsme pro náš systém zvolili zatím poslední vydanou verzi, kterou je Oxygen.

4 Rozšíření pro veřejné knihovny

Jak už jsme si několikrát naznačili, náš systém by se dal použít i ve veřejných knihovnách. Na závěr naší práce si tedy rozebereme, jakým způsobem bychom mohli náš systém obohatit o některé další funkce a tím zaujmout i uživatele nejen domácích knihoven. Samozřejmě by se dalo najít spoustu dalších rozšíření, ale to není cíl naší práce.

Největší změny by mohly nastat v serverové a klientské aplikaci. Ve velkých knihovnách se předpokládá, že proces katalogizace nebude práce pouze jednoho zaměstnance. Je docela pravděpodobné, že jeden zaměstnanec bude mít za úkol skenování knih, zatímco jiný bude tyto knihy dále zpracovávat. Dále zde přibydou uživatelé v roli návštěvníků knihovny. Velký rozdíl při využití systému mezi domácí a veřejnou knihovnou je ten, že nebude nabízet své funkce pouze jednomu uživateli.

Jednou z výhod návrhu našeho systému je rozdělení serverové části a klientské aplikace. Touto výhodou je, že se obě aplikace nebudou vzájemně omezovat. Serverová část může přijímat data z Android zařízení a nebude tím zpomalovat chod klientské aplikace. Zatímco v domácím prostředí budou nejspíše obě aplikace spuštěné v lokální síti, ve veřejných knihovnách tomu bude jinak. Jelikož potřebujeme, aby byl zajištěn přístup k systému i ze strany návštěvníků knihovny, bude potřeba alespoň klientskou aplikaci nasadit na webhosting. S největší pravděpodobností by serverová aplikace mohla zůstat běžet pouze v lokální síti, protože není důvod k ní přistupovat odjinud.

4.1 Migrace databáze

Nyní se dostáváme k problému s použitím databáze SQLite. Jak již bylo popsáno, jedná se o databázi, která je obsažena pouze v jednom lokálně uloženém souboru. To nám v domácím prostředí, kde bude serverová i klientská část běžet na jednom stroji, vyhovuje. V případě veřejné knihovny je to však problém, protože tyto dvě části budou od sebe odděleny. Je jasné, že budeme potřebovat jinou databázi. Jelikož se jedná o knihovny, není předpoklad nějaké velké zátěže databázového systému. Nejspíše tedy půjde o nějaký cenově dostupný relační databázový systém například MySQL. Náš systém v tomto případě se změnou databáze počítá už od doby návrhu.

Ke zjednodušení změny velmi pomůže již naimplementovaný přístup k databázi pomocí frameworku Hibernate. Pro změnu databázového systému bude důležité importovat do serverové i klientské aplikace potřebný JDBC driver k příslušné databázi. Dále bude potřeba změnit u obou konfigurační soubor Hibernate, v našem případě jde o hibernate.cfg.xml. Zde změníme několik věcí. Jednou z nich je položka connection.driver_class, kde uvedeme právě driver nové databáze. V řádku connection.url napíšeme adresu vzdáleného serveru, na kterém databázový systém běží. Následují další dvě důležité položky nastavení, kterými jsou uživatelské jméno a heslo pro přístup k databázi. Po těchto změnách bude systém schopný spolupracovat s jinou databází než je SQLite. Pozitivní na tom je, že chceme-li změnit typ databáze, všechno se to děje na úrovni konfiguračního souboru Hibernate, a tudíž nemusíme vůbec nic měnit ve zdrojovém kódu logiky celého systému.

4.2 Změny v Android aplikaci

Když jsme si uvedli, jaké změny nastanou v nasazení systému a také jak jednoduchá je migrace na jiný databázový systém, je čas si uvědomit další požadavky, které budou kladeny

na systém ve veřejné knihovně na rozdíl od té domácí. V domácí knihovně předpokládáme, že uživatel má každou knihu pouze jednou, a pokud jich má náhodou víc, není pro něj podstatné, jestli má tuto informaci ve své evidenci. To však není případ veřejných knihoven, ve kterých je tato informace důležitá. Knihovny totiž musí znát počet každé knihy, aby měly přehled o tom, kolik jich mohou ve stejnou dobu vypůjčit. Tím jsme naznačili, že bude žádoucí v aplikaci pro Android doplnit možnost zadání počtu knih při skenování.

4.3 Rozšíření klientské aplikace

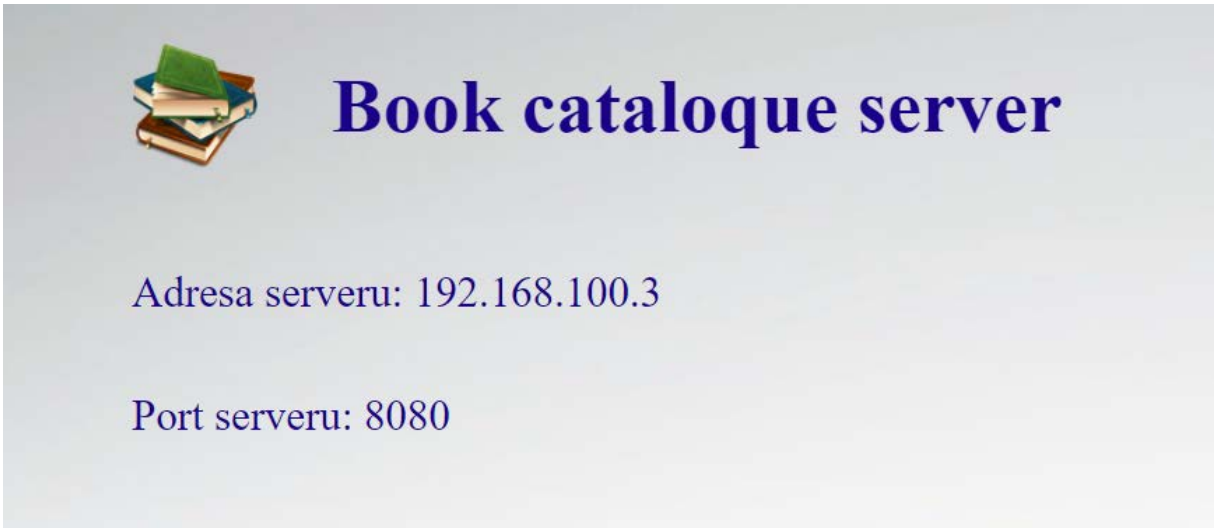
Hlavní změny musí proběhnout v klientské aplikaci. V tento okamžik, jako uživatel našeho systému, máme přístup ke všem knihám a můžeme je jednoduše přidávat, editovat nebo mazat. Zřejmě je nám jasné, že takový přístup do systému není žádoucí pro návštěvníky knihoven. Nicméně i tak můžeme toho, jak je systém implementován, využít, pouze musíme práva k těmto funkcím nechat zaměstnancům dané knihovny.

Z tohoto důvodu by zde jako první měla být doimplementovaná autentizace uživatelů, abychom oddělili roli zaměstnance od registrovaného uživatele a návštěvníka. Zaměstnanec by tedy mohl po úspěšné autorizaci s katalogem knih manipulovat. Tak jako v jiných odvětvích i v knihovnách mají zaměstnanci různé pozice, podle toho by se měla omezit také práva u zaměstnanců. Nebylo by nejspíš příliš vhodné, aby každý zaměstnanec mohl libovolně knihy mazat.

Nepřihlášeným návštěvníkům stránek by měl zůstat viditelný pouze obsah katalogu knih, rozhodně by neměli mít možnost cokoliv v systému měnit. Za účelem vypůjčení knih by se na úvodní webové stránce měla objevit možnost přihlášení uživatelům, kteří jsou v dané knihovně registrovaní. Po přihlášení by měl takový uživatel získat další možnosti, jak se systémem pracovat. V tomto okamžiku by bylo vhodné, aby došlo k propojení se systémy, které jsou běžně používané v knihovnách. Další rozšíření našeho systému by totiž znamenalo pouze implementaci toho, co se už používá.

5 Ukázky systému

Obr. 9.: Ukázka Android aplikace



Obr. 10.: Ukázka serverové části

Obr. 11.: Ukázka klientské aplikace

ZÁVĚR

Cílem této práce bylo vytvořit systém, který by usnadnil katalogizaci domácího knižního fondu. Usnadnění spočívá v možnosti přidávat knihy do katalogu pomocí mobilních zařízení operujících na platformě Android a v následném vyhledávání informací o knihách z externích zdrojů.

Velmi obohacující pro mě byl především vývoj aplikace pro Android, jelikož jsem se s ním do doby této práce nikdy nesetkal. Práce s ním vykazovala hlavně ze začátku značné potíže, ale ty se mi později podařilo úspěšně vyřešit. Šlo především o problémy spojené s kompatibilitou mezi různými verzemi systému. Jednalo se o části spojené s pořizováním snímku a jejich následnému zaznamenání do externí paměti. U novějších verzí Android se přiřazují práva pro přístup do úložiště až za běhu aplikace, tedy tím že se aplikace ptá uživatele, zda jí povolí tento přístup. Ve starších verzích stačilo, aby tato práva byla zapsána v souboru AndroidManifest.xml.

Při řešení problému získávání klíčových informací z knih, které bylo vyvíjeno pro systém Android, jsem byl velmi mile překvapen z technologie Mobile Vision API, kterou lze používat k detekci obličejů, textu nebo čárových kódu z obrázků. V této práci je použita pouze část věnující se rozpoznávání textu, ale pokud by byla v budoucnu příležitost, rád bych si vyzkoušel i ostatní možnosti, které nám tato technologie nabízí.

Další pozitivní zkušenosti jsem získal při vývoji klientské aplikace. Jelikož jsem se s vývojem webových aplikací setkal především v jiných jazycích než Java, a to především v jazyce C# .NET s pomocí frameworku ASP.NET MVC, byla pro mě příjemným překvapením práce s technologií JavaServer Faces, ve které je vývoj webových aplikací založený na podobném principu jako v technologii ASP.NET MVC.

Výsledná aplikace tedy umožňuje skenování knih prostřednictvím Android zařízení, při kterém dokáže identifikovat ISBN. To spolu s pořízenými snímky posílá na server, který záznam o naskenované knize ukládá do databáze. V klientské aplikaci jsou knihy na základě ISBN dohledány z webů knihoven nebo zvlášť doplněny uživatelem.

Možným rozšířením systému jsem se zabýval už od počátku jeho vývoje. Díky tomu, že na začátku došlo z mé strany k nedorozumění v pochopení zadání práce, kdy jsem si představoval spíše systém pro veřejné knihovny než pro domácí využití, bylo pro mě cílem připravit systém tak, aby byla možnost ho v budoucnu co nejsnadněji rozšířit právě pro veřejné knihovny.

LITERATURA

[1] LUTONSKÝ, Martin. *Evidence LSoft* [online]. Brno: LSoft.cz, 2018 [cit. 2018-04-10]. Dostupné z: http://www.lsoft.cz/

[2] HOOGERDIJK, Alwin. *Book Collector* [online]. Nizozemí: Collectorz.com, 2018 [cit. 2018-04-26]. Dostupné z: https://www.collectorz.com/book/book-collector

[3] *Mobile vision* [online]. c2016 [cit. 2018-04-17]. Dostupné z: https://developers.google.com/vision/

[4] *Face Detection Concepts Overview* [online]. c2016 [cit. 2018-04-17]. Dostupné z: https://developers.google.com/vision/face-detection-concepts

[5] *Barcode API Overview* [online]. c2017 [cit. 2018-04-17]. Dostupné z: https://developers.google.com/vision/android/barcodes-overview

[6] *Text Recognition API Overview* [online]. c2017 [cit. 2018-04-17]. Dostupné z: https://developers.google.com/vision/android/text-overview

[7] *JavaServer Faces.org* [online]. USA: Oracle, 2012 [cit. 2018-04-17]. Dostupné z: http://www.javaserverfaces.org/

[8] *JavaServer Faces Technology Overview* [online]. USA: Oracle, 2018 [cit. 2018-04-17]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/overview-140548.html

[9] *JSF - Overview* [online]. India: Tutorials Point, 2018 [cit. 2018-04-17]. Dostupné z: https://www.tutorialspoint.com/jsf/jsf_quick_guide.htm

[10] Representational state transfer. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2018 [cit. 2018-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Representational_state_transfer

[11] BURKE, Bill. *RESTful Java with JAX-RS 2.0: Designing and developing distributed web services*. 2. vyd. USA: O´REILLY Media, 2013. ISBN 978-1449361341.

[12] SUPOL, Jan. *Jersey: RESTful Web Services in Java.* [online]. USA: Oracle, 2018 [cit. 2018-04-28]. Dostupné z: https://jersey.github.io/

[13] HEDLEY, Jonathan. *Jsoup: Java HTML Parser* [online]. c2018 [cit. 2018-04-17]. Dostupné z: https://jsoup.org/

[14] LE HÉGARET, Philippe a kol. *What is the Document Object Model?* [online]. USA: W3C, 2000 [cit. 2018-04-25]. Dostupné z: https://www.w3.org/TR/DOM-Level-2-Core/introduction.html

[15] Document Object Model. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2018 [cit. 2018-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Document_Object_Model

[16] BRITTAIN, Jason a Ian F. DARWIN. *Tomcat: The Definitive Guide*. 2. vyd. USA: O´REILLY Media, 2008. ISBN 978-0-596-10106-0.

[17] BAUER, Christian a Gavin KING. *Java Persistence with Hibernate: Revised edition of hibernate in action*. USA: Manning Publications Co., 2007. ISBN 1-932394-88-5.

[18] KING, Gavin a kol. *Hibernate Getting Started Guide. Preface* [online]. USA: Red Hat, 2012 [cit. 2018-04-24]. Dostupné z: https://docs.jboss.org/hibernate/orm/3.6/quickstart/en-US/html_single/

[19] *HIBERNATE* [online]. USA: Red Hat [cit. 2018-04-17]. Dostupné z: http://hibernate.org/orm/

[20] *Appropriate Uses For SQLite* [online]. c2018 [cit. 2018-04-17]. Dostupné z: https://www.sqlite.org/whentouse.html

[21] PISTORIUS, Vladimír. *Jak se dělá kniha: Příručka pro nakladatele*. 3. vyd. Příbram: Pistorius & Olšanská, 2011. ISBN 978-80-87053-50-8.

[22] *ANDROID: Vývoj aplikací pro pokročilé i začátečníky* [online]. Praha: GOPAS, 2018 [cit. 2018-04-25]. Dostupné z: https://www.gopas.cz/News/ANDROID--Vyvoj-aplikaci-pro-pokrocile-i-zacatecniky.aspx

[23] FIALA, Daniel. *Databáze knih: Váš knižní svět* [online]. Polná: Databazeknih.cz, 2018 [cit. 2018-04-26]. Dostupné z: https://www.databazeknih.cz/

[24] *Meet Android Studio* [online]. c2018 [cit. 2018-04-19]. Dostupné z: https://developer.android.com/studio/intro/index.html

[25] Android Studio. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, c2001-2018 [cit. 2018-04-19]. Dostupné z: https://en.wikipedia.org/wiki/Android_Studio

[26] ISTRIA, Mickael. *The Official Eclipse FAQs* [online]. Canada: Eclipse Foundation, 2016 [cit. 2018-04-19]. Dostupné z: https://wiki.eclipse.org/The_Official_Eclipse_FAQs

[27] Eclipse (software). *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2018 [cit. 2018-04-19]. Dostupné z: https://en.wikipedia.org/wiki/Eclipse_(software)

SEZNAM PŘÍLOH

K práci je přiložen CD disk, který obsahuje elektronickou verzi tohoto dokumentu ve formátu PDF a zdrojové kódy systému.

- Adresář Text - text bakalářské práce
- Adresář Android - Android aplikace
- Adresář Server - Serverová část systému
- Adresář Klientská aplikace - Webová aplikace